# Abschlussbericht des Projekts
# “Lokalisierung mit Bluetooth LE”

Björn Seifert

Knowledge & Data Engineering Group
University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
http://www.kde.cs.uni-kassel.de/seifert

## 1 Motivation

Im Rahmen des VENUS-Projekts wurde bisher die raumgenaue Lokalisierung von Konferenzteilnehmern mit Hilfe von aktiven RFID-Tags und zugehörigen RFID-Empfängern in den Konferenzräumen durchgeführt [SDA+11]. Hierbei sendet das RFID-Tag jedes Teilnehmers eine Kennung aus, die von den Antennen der einzelnen Empfänger in verschiedenen Empfangsstärken und Häufigkeiten empfangen und an einen zentralen Server übermitteln wird. Aus diesen gesammelten Daten errechnet der Server die wahrscheinliche Position des Teilnehmers am Konferenzort. Diese Technik erfordert eine eine funktionierende vorhandene oder neu aufzubauende LAN-Netzwerkinfrastruktur am Veranstaltungsort sowie einen administrativen Aufwand beim Verteilen der RFID-Tags an die Konferenzteilnehmer.

Im Rahmen dieser Projektarbeit soll die Möglichkeit evaluiert werden, dieses System auf “Bluetooth Low Energy” (Bluetooth LE) umzustellen, bei dem vorhandene Smartphones der Teilnehmer die Signale von mehreren stationär in den Konferenzräumen angebrachten Bluetooth-Sendern, sogenannten Bluetooth LE-Beacons (von Apple unter dem Namen “iBeacon” vermarktet), empfangen und daraus ihre eigene Position im Raum ermitteln. Somit kann sowohl auf die Netzwerkinfrastruktur am Konferenzort verzichtet werden, als auch durch die Nutzung von vorhandenen Smartphones der Teilnehmer sowie günstiger Bluetooth-Sender der Kostenaufwand für den Aufbau deutlich reduziert werden.

## 2 Zusammenfassung des Themas

Basierend auf der vom AltBeacon-Projekt `http://altbeacon.org/` veröffentlichten Android-Library zum Erkennen von Bluetooth-Beacons nach der AltBeacon-Spezifikation (die kompatibel zur iBeacon-Spezifikation und den kommerziell verfügbaren Beacons der Firma Estimote ist) wurde im Laufe des Projekts ein Android-Framework und eine zugehörige Beispiel-App entwickelt, die die Announcements von einem oder mehreren Beacons empfängt und daraus die aktuelle Position des Telefons auf dem abgedeckten Gebiet ermittelt.

Hierbei wurden verschiedene Methoden der Positionsberechnung durch Fingerprinting evaluiert sowie ein Vergleich zu einem Ansatz mit Trilateration gezogen.

## 3 Die Bluetooth Low Energy - Technik

Einen umfassenden Überblick in die Technik geben die Autoren Gomez, Oller und Paradells im Sensors-Magazin [GOP12]. Bluetooth LE ist seit 2010 Teil der Bluetooth 4.0 Spezifikation, es sendet wie viele weitere Anwendungen (WLAN, Babyphone, Funkkopfhörer) im lizensfreien ISM-Band (Industrial, Scientific and Medical Band) bei 2.4GHz. Bei der Spezifizierung wurde besonders Augenmerk auf einen geringen Energieverbrauch gelegt, so dass eine CR2477-Knopfzelle (1000mAh) eine theoretische Laufzeit von bis zu 14 Jahren erlauben sollte, die in der praktischen Anwendung immerhin noch bei 1-2 Jahren liegt.

Bluetooth LE wird wegen seiner geringen Energieanforderungen inzwischen als häufigstes Protokoll für Einsatzzwecke benutzt, bei denen regelmäßig kleine unidirektionale Datenpakete verschickt werden müssen. Als Beispiel kann man Pulsmesser, Schlaf- und Aktivitätstracker oder Smart-Home-Anwendungen wie Temperatursensoren nennen. Da diese Einsatzwecke Teil des aufkommenden sogenannten "Internets der Dinge" ist, sind heute auch alle aktuellen Smartphones unter Android (ab dem Nexus 4) und iOS (ab dem iPhone 4s) mit LE-fähigen Bluetoothmodulen ausgestattet.

### 3.1 iBeacon

Dem Bluetooth-Funkmodul stehen 40 Kanäle zur Verfügung, wobei davon drei Kanäle ausschließlich sogenannte "advertising channels" sind, die nicht für die Übertragung von Nutzdaten sondern zum Management des Protokolls (wie z.B. Verbindungsaufbau) genutzt werden oder um angebotene Dienste mitzuteilen. Eine weitere Anwendung der "advertising channels" ist das in diesem Projekt benutzte "Broadcasting".

Im Jahr 2013 wurde das von Apple entwickelte und standardisierte "iBeacon"-Protokoll [Inc] vorgestellt. Dabei werden in regelmäßigen, konfigurierbaren Abständen im Sekundenbereich Datenpakete als Funkfeuer (engl. "Beacon") verschickt, die im Wesentlichen aus einer Kennnummer (UUID), zwei benutzerdefinierten Werten sowie dem Hinweis auf die verwendete Sendeleistung bestehen. Diese Funksignale haben unter Idealbedingungen eine Reichweite von bis zu 70 Metern und werden in der Praxis dazu genutzt, die Nähe zu einer bestimmten, mit einem Sender ausgestatteten Position festzustellen. Anwendungsgebiete sind in der Lagerdisposition (Auffinden und Verfolgen von Warenbeständen) sowie in der kontextsensitiven Werbung zu suchen, dabei wird meistens aus der empfangenen Sendestärke nur eine grobe Einteilung der Entfernung des Empfängers zum Sender vorgenommen, typischerweise in der Auflösung "Immediate" (wenige cm), "Near" (wenige m) und "Far" (ab 10m).

iBeacons werden in verschiedenen Ausführungen und von verschiedenen Firmen zu einem Preis um die 30 Dollar das Stück angeboten. Meist wird dazu eine Smartphone-App angeboten, um sich per Bluetooth (nicht LE) verbinden zu können um die Sendeparamter zu konfigurieren. Bei den hier im Experiment verwendeten iBeacons der Firma estimote sind das Sendeleistung (in Milliwatt) und -häufigkeit (1-100Hz) sowie die beiden benutzerdefinierten 2-Byte-Werte

“Major” und “Minor”, die man als Erkennungszeichen benutzen kann. So kann man je nach Einsatzzweck zum Beispiel definieren, dass eine bestimmter Veranstaltungsort (oder ein Warenlager) eine “Major”-Nummer zugewiesen bekommt und dort das einzelne Beacon über die “Minor”-Nummer unterschieden wird. Die dritte ausgesendete Nummer, die UUID, ist bei allen iBeacons ab Werk die gleiche und die einzige, die von einem iPhone als gültig akzeptiert wird.

### 3.2 USB-Bluetoothmodul als Alternative

Als preiswertere Alternative zu den komerziellen iBeacons ist es mit den meisten Bluetooth 4.0 - USB-Sticks möglich, einen iBeacon für unter 10 Euro zu simulieren. Dafür kann man mit einem Kommandozeilentool unter Linux den Stick in einen “raw”-Modus schalten, in dem er beliebige von Benutzer übergebene Pakete regelmäßig als Broadcast verschickt.

Nachdem der Stick korrekt konfiguriert und die Übertragung gestartet wurde, sendet er die Pakete auch ohne Verbindung zum PC weiter, solange die Stromzufuhr nicht unterbrochen wird. Dies lässt sich durch die Verwendung eines stromgebenden USB-Hubs oder das Anlöten einer 5-6V-Spannungsversorgung an die entsprechenden USB-Pins erreichen.

### 3.3 Im Projekt verwendete Hardware-Konfiguration

Für die Lokalisierungsexperimente benutze ich ein Google Nexus 4 vom Hersteller LG mit Android 4.3 als Empfänger, auf Senderseite kommt ein iBeacons der Firma estimote sowie drei Bluetooth 4.0 USB-Sticks zum Einsatz. Das iBeacon sendet mit voller Leistung und ebenfalls wie die USB-Sticks mit 1Hz Paketfrequenz.

## 4 Verwendete Android-Software

### 4.1 AltBeacon-Library

Zur Kommunikation mit dem Bluetooth-Chip des Nexus 4 verwende ich die AltBeacon-Library. Diese startete unter dem Namen “iBeacon for Android” als aus Apple-Bibliotheken reverse-engineertes Produkt der Firma Radius Networks, das jedoch im Juli 2014 auf Grund juristischen Drucks von Apple eingestellt werden musste [Wal].

Daraufhin spezifizerte die Firma Radius Networks ihr eigenes, quelloffenes sogenanntes “AltBeacon”-Protokoll und entwickelt seitdem eine zugehörige OpenSource-Library auf GitHub. Diese ist noch immer in der Lage, ebenfalls Apple-zertifizerte iBeacons zu erkennen, was ich mir in diesem Projekt zu nutzen mache.

### 4.2 SQLite-Framework OrmLite

Android benutzt zur Speicherung von Programmdaten neben key-value-Stores für Konfigurationsparameter ausgiebig SQLite als relationale Datenbank für größere Datenmengen. Um in dieser auf einfache Art und Weise Java-Objekte speichern zu können, verwende ich "Object Relational Mapping" (ORM) mit Hilfe der Java-Bibliothek "OrmLite". Diese generiert durch simple Java-Annotationen Programmcode zum Speichern und Auslesen von Java-Objekten in eine relationale Datenbank. So ist zum Beispiel die Klasse **Messung** wie folgt implementiert:

```
public class Messung{
    @DatabaseField(generatedId = true)
    private Integer id;

    @DatabaseField
    public Date zeitstempel;

    @DatabaseField
    public Boolean learned = false;

    @DatabaseField(foreign = true)
    public Ort ort;

    @ForeignCollectionField(eager = false)
    ForeignCollection<Sichtung> sichtungen;
}
```

Das gesamte Datenbankschema ist in Abbildung 1 dargestellt. Jede Sekunde wird vom Telefon eine **Messung** durchgeführt, die je nach Anzahl der empfangenen Beacons einer Anzahl an aktuellen **Sichtungen** zugeordnet wird. Eine **Sichtung** besteht aus einem (unveränderlichem) **iBeacon** sowie der gemessenen Empfangsstärke (rssi). In jeder **Messung** wird zudem festgehalten, ob diese zum Anlernen des Models benutzt werden soll und wenn ja, an welchem **Ort** sie stattfand. Ansonsten wird später der klassifizerte **Ort** in der **Messung** gespeichert.

In dem **iBeacon**-Objekt wird zudem noch die Sendestärke (power) gespeichert. Diese wird von den Beacon-Herstellern angegeben als Empfangsstärke (rssi), mit dem das Beacon von einem iPhone in 1m Entfernung erkannt wird.

### 4.3 Machine-Learning-Bibliothek Weka

Im Experiment sollen verschiedene Methoden der Positionsberechnung mit Hilfe eines Fingerprinting-Ansatzes evaluiert werden, unter anderem die im RFID-Paper [SDA$^+$11] beschriebenen Klassifikatoren Naive Bayes [Mit97], K-Nearest Neighbour [Mit97]und Random Forrest [Bre01], zusätzlich Multiple Linear Regression [Fre05]. All diese Algorithmen sind in der Java-Software Weka [WF99] bereits implementiert. Mit Hilfe der Arbeit des Software Engineers Rj Marsan [Mar] (Angestellter bei Google), der die Weka-Bibliotheken nach Android portiert hat, konnten die Klassifikatoren direkt unter Android eingesetzt werden.

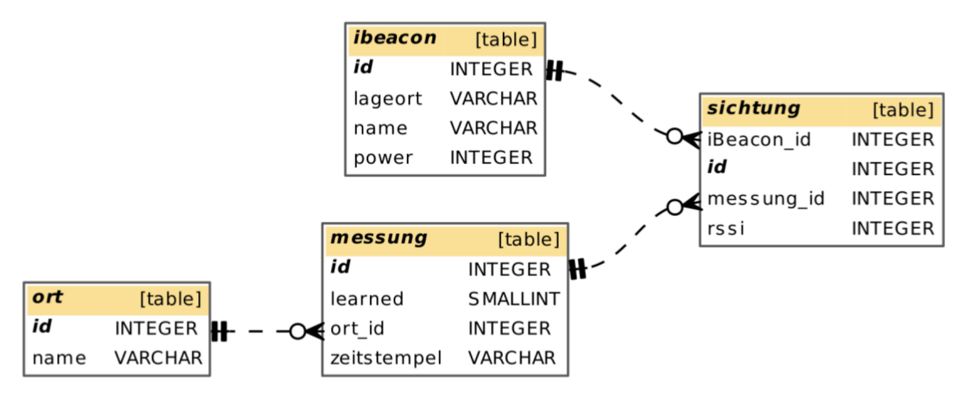



Abb. 1: Datenbankschema

# 5 Android-App

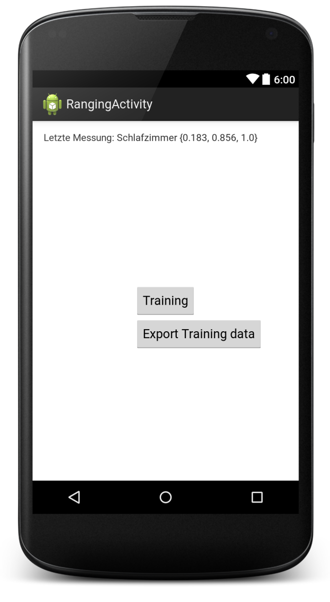
Die entwickelte Android-App verwendet die AltBeacon-Library zum Messen der Bluetooth-LE-Stärken und läuft ab Andorid 4.3 Jelly Bean (SDK-Version 18). Sie besteht im Grunde nur aus zwei Bildschirmmasken (¸emphActivities). Der Startbildschirm zeigt die aktuelle Bluetooth-Messung an und zeigt wenn möglich die Ausgabe des Klassifikators. Außerdem bietet er die Möglichkeit, das komplette Modell inkl. aller Messdaten auf die Speicherkarte zu exportien.
Die von Startbildschirm mit dem Knopf *Training* erreichbare zweite Activity bietet zum einen das Hinzufügen von Messorten und Starten der Referenzmessungen "Position lernen" sowie eine Schaltfläche "Update Classifier" zum Neuberechnen des Modells, siehe Abbildung 2b.
Die Messung wird beim Lernen in der oben angegebenen Art in einer Datenbank hinterlegt, das Modell liegt nach der Erzeugung ebenfalls als Objekt vor und wird serialisiert als *Shared Preference* im Datenspeicher der App persistent gespeichert, um nach Neustarts ohne Neuerzeigung wieder ausgelesen werden zu können.

# 6 Durchgeführte Experimente

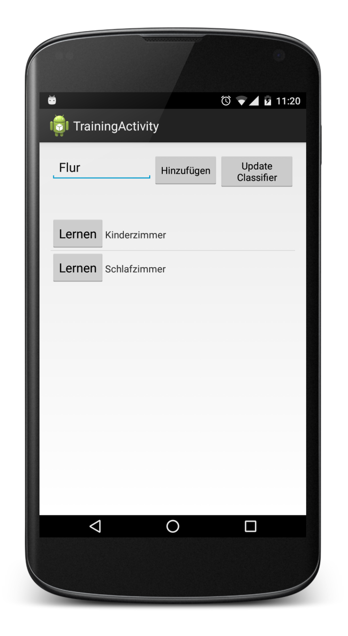
Der Aufbau der Experimente fand in meiner Wohnung in Friedland statt, in Abbildung 3 ist die Lage der Beacons und der zu unterscheidenen Messpunkte dargestellt. Der Sender im Schlafzimmer ist ein iBeacon von Estimote, die restlichen Beacons sind durch USB-Sticks realisiert. Insgesamt waren 4 Beacons verteilt, die Wohnung wurde sinnvoll in 10 Messpunkte unterteilt, die sich an markanten Einrichtungsgegenständen orientierten.

## 6.1 Aufnahme der Messungen und Auswertung auf dem Gerät

An den 10 Messpunkten wurden jeweils für 30 Sekunden Referenzwerte aufgenommen, was 30 Messungen der empfangenen Signalstärken der jeweils sichbaren




(a) Anzeige der Messung




(b) Lernen des Modells

Abb. 2: Bildschirme der Android-App

Beacons entspricht. Aus den so gewonnen 300 Messungen wurde danach ein Modell berechnet, wobei für die verschiedenen noch zu erläuterenden Klassifikatoren jeweils der Quelltext der App neu angepasst und übersetzt werden musste, da eine Auswahl des Klassifikators innerhalb der App nicht möglich ist.

Danach bin ich mit der App durch die Wohnung gelaufen und habe beobachtet, wie nun die tatsächliche Position automatisch den vermuteten Messpunkten zugeordnet wurde. Diese rein subjektive Auswertung sah bereits recht vielversprechend aus, die genaue Analyse der Vorhersage-Genauigkeit wurde jedoch offline mit Hilfe der auf dem Handy gewonnen Messdaten innerhalb von Weka auf dem Desktop durchgeführt.

### 6.2 Vergleich der unterschiedlichen Klassifikatoren

Es wurden vier verschiedene Klassifikatoren für das Zuordnen der Messungen zu den Messpunkten untersucht:

- k-Nearest-Neighbours ($k = 5$)
- Logistic regression
- Naive Bayes

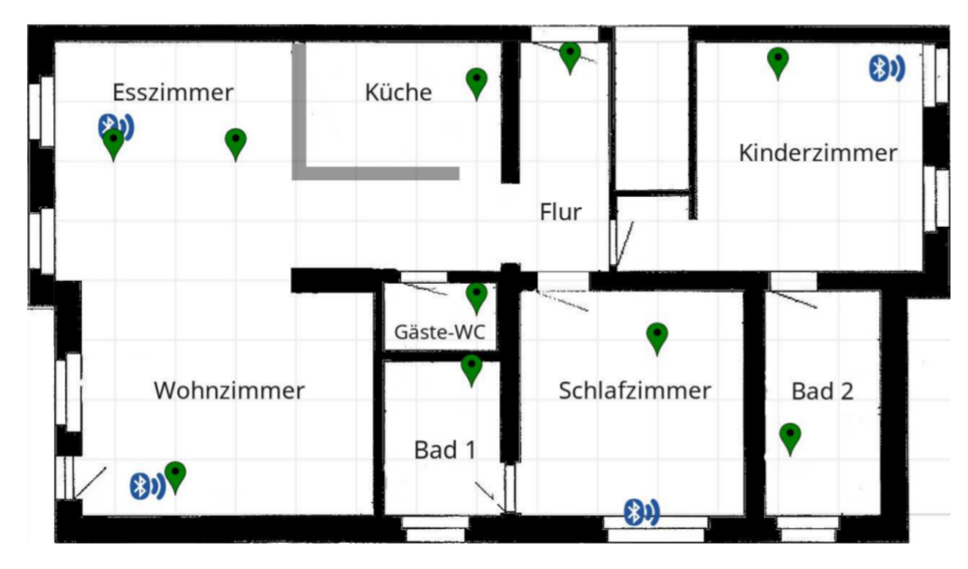



Abb. 3: Lage der iBeacons und Messpunkte

– Random Forrest

Die Auswahl der Klassifikatoren geschah mit dem Hintergrund, dass die in Weka gegebene Implementierung einfach eingesetzt werden konnte und die Modelle schnell auf einem Telefon berechnet werden können. Es sollen zwei Algorithmen exemplarisch kurz erleutert werden:

**k-Nearest-Neighbours** Die kNN-Klassifikation weist jeden zu bewertenden Messwert das Ergebnis zu, das die meisten seiner (hier fünf) geometrischen Nachbar-Referenzmesspunkte aufweisen. Dazu wird ein n-dimensionaler Raum aufgespannt, wobei n die Gesamtanzahl der Beacons ist. Jeder rssi-Wert beschreibt einen Koordinatenabschnitt, wobei ein in einer Messung nicht gesehenes Beacon dort einen Wert von 0 entspricht. Für die ermittlung der Entfernung zweier Messpunkte wird in dieser Anwendung die Manhattan-Distanz verwendet, welche die Abstände für jede Richtung des Raumes addiert.

**Random Forrest** Bei der Random Forrest-Methode werden (hier 10) Entscheidungsbäume aus jeweils zufällig ausgewählten Referenzmessugen aufgebaut. Jede zu bewertende Messung wird nun parallel nach allen Entscheidungsbäumen klassifiziert, wobei am Schluss eine Mehrheitsentscheidung der Einzellklassifikationen die Gesamtklassifikation ergibt.

Für den objektiven Vergleich der Klassifikatoren wurden für jeden Messpunkt der Wohnung mit Hilfe der Android-App weitere 170 Messungen durchgeführt, so dass eine Datenmenge von insgesamt 2000 Messungen zur Verfügung stand. Diese

wurden mit einem 10-fachen Kreuzvalidierungsverfahren den Klassifikatoren angeboten. Die Ergebnisse sind in den Abbildungen 4 - 5 dargestellt. Sie sind wie folgt zu lesen: auf der linken Einteilung kann man die tatsächliche Position während der Messung ablesen, während oben das Ergebniss der Klassifikation dargestellt ist. Die Diagonalen der Korrekt erkannten Messungen sind grün hinterlegt während ein gelber Hintergrund ausdrückt, dass hier eine Fehlklassifikation von mehr als 10 Prozent der Messungen vorliegt.

Correctly Classified Instances: 62% (k=5)

| | Esszimmer links | Esszimmer rechts | Wohnzimmer | Küche | Gäste-WC | Bad 1 | Kinderzimmer | Flur | Schafzimmer | Bad 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Esszimmer links** | 73% | 15% | | | | | | | | |
| **Esszimmer rechts** | | 58% | 8% | | 28% | | | | | |
| **Wohnzimmer** | 12% | 10% | 70% | | 7% | | | | | |
| **Küche** | | | | 77% | 10% | | | 10% | | |
| **Gäste-WC** | | | | 11% | 76% | 11% | | | | |
| **Bad 1** | | | | | 35% | 42% | | 12% | 6% | |
| **Kinderzimmer** | | | | | | | 90% | | | |
| **Flur** | | | | 6% | | | 11% | 72% | 8% | |
| **Schafzimmer** | | | | | | | | 33% | 56% | |
| **Bad 2** | | | | | 11% | 8% | 11% | 44% | 14% | 8% |

k-Nearest Neighbours

Correctly Classified Instances: 69%

| | Esszimmer links | Esszimmer rechts | Wohnzimmer | Küche | Gäste-WC | Bad 1 | Kinderzimmer | Flur | Schafzimmer | Bad 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Esszimmer links** | 88% | | 8% | | | | | | | |
| **Esszimmer rechts** | 9% | 51% | 23% | | 15% | | | | | |
| **Wohnzimmer** | | 20% | 78% | | | | | | | |
| **Küche** | | 7% | | 80% | | | | | | |
| **Gäste-WC** | | | | 13% | 67% | 14% | | | | |
| **Bad 1** | | | | | 23% | 56% | | | 8% | |
| **Kinderzimmer** | | | | | | | 87% | | 9% | |
| **Flur** | | | | 8% | | | 9% | 60% | 6% | 15% |
| **Schafzimmer** | | | | | | | | 28% | 51% | 18% |
| **Bad 2** | | | | | 5% | | | 15% | 8% | 64% |

Logistic Regression

Abb. 4

Der kNN-Algorithmus kann zwar viele Positionen gut unterscheiden, versagt jedoch beim Erkennen des Bad 2. Die durch Einrichtungsgegenstände, WÄnde und Körperabschirmung gestörten Messsignale scheinen die empfangenen Signalstärke so zu stören, dass keine geometrische Kontinuität mehr gegeben ist. Dies ergibt auch, dass eine Positionsbestimmung durch Trilateration sinnlos ist, da hier ebenfalls eine geometrische Betrachtung der Messwerte gefordert wäre.
Logistic Regression und Naive Bayes schneiden etwas besser ab, während sich zeigt, dass der Random Forrest-Klassifikator mit insgesamt 76 Prozent richtig zugeordneten Messungen die besten Ergebnisse erreicht und einzig zwischen Gäste-WC und Bad 1 nur schlecht entscheiden kann.

## 7 Fazit

Die Auswahl eines geeigneten Klassifikators für die Positionserkennung per Bluetooth-LE ist ziemlich eindeutig auf Random Forrest gefallen, welcher auch sehr schnell von einem Mittelklasse-Mobilfunktelefon berechnet werden kann. Die Implementierung der App war trotz verwendung der bereits vorhandenen Weka-Algorithmen sehr aufwändig, da diese Bibliothek eine für mobile Anwendungen ungewohnt komplizierte Datenstruktur als Eingabe erfordert. Deshalb blieben kaum Kapazitäten übrig, die Android-App über die wichtigste Funktionalität, dem Ausnehmen und der Auswertung der Messungen hinaus ansprechend zu

Correctly Classified Instances: 71%

| | Esszimmer links | Esszimmer rechts | Wohnzimmer | Küche | Gäste-WC | Bad 1 | Kinderzimmer | Flur | Schafzimmer | Bad 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Esszimmer links** | 88% | 8% | | | | | | | | |
| **Esszimmer rechts** | 8% | 65% | 14% | | 8% | | | | | |
| **Wohnzimmer** | | 10% | 88% | | | | | | | |
| **Küche** | | | | 80% | | | | | | |
| **Gäste-WC** | | 8% | | 11% | 73% | 8% | | | | |
| **Bad 1** | | | | 8% | 33% | 40% | | 8% | | 6% |
| **Kinderzimmer** | | | | | | | 88% | | | 9% |
| **Flur** | | | | 6% | | | 8% | 69% | 12% | |
| **Schafzimmer** | | | | | | | | 25% | 48% | 26% |
| **Bad 2** | | | | | | | | 8% | 22% | 63% |

Naive Bayes

Correctly Classified Instances: 76% (10 Bäume)

| | Esszimmer links | Esszimmer rechts | Wohnzimmer | Küche | Gäste-WC | Bad 1 | Kinderzimmer | Flur | Schafzimmer | Bad 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Esszimmer links** | 86% | 6% | 8% | | | | | | | |
| **Esszimmer rechts** | | 78% | | | 6% | | | | | |
| **Wohnzimmer** | | 10% | 83% | | | | | | | |
| **Küche** | | | | 75% | | | | 7% | | |
| **Gäste-WC** | | | | 8% | 62% | 24% | | | | |
| **Bad 1** | | | | | 23% | 58% | | | | 6% |
| **Kinderzimmer** | | | | | | | 88% | | | 9% |
| **Flur** | | | | | | | 9% | 65% | | 20% |
| **Schafzimmer** | | | | | | | | | 79% | 15% |
| **Bad 2** | | | | | | | | | 8% | 78% |

Random Forrest

Abb. 5

gestalten.
Die mit nur vier Beacons aufgenommenen Ergebnisse sind meiner Meinung nach wirklich aussagekräftig und die Kosten des Aufbaus halten sich mit 50-150 Euro für die Abdeckung einer Fläche von knapp 100 Quadratmetern sehr in Grenzen.

Der Quelltext der Android-App kann in einem Github-Repository eingesehen werden.

## Literatur


Bre01. Breiman, Leo: *Random Forests.* Machine Learning, 45:5–32, 2001.

Fre05. Freedman, David: *Statistical Models : Theory and Practice.* Cambridge University Press, August 2005.

GOP12. Gomez, Carles, Joaquim Oller und Josep Paradells: *Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology.* Sensors, 12(9):11734–11753, 2012.

Inc. Inc., Apple: *iBeacon for Developers.* `https://developer.apple.com/ibeacon/`. Accessed: 2015-06-01.

Mar. Marsan, Rj: *Weka for Android.* `https://github.com/rjmarsan/Weka-for-Android`. Accessed: 2015-06-01.

Mit97. Mitchell, T.: *Machine Learning.* McGraw Hill, 1997.

SDA+11. Scholz, Christoph, Stephan Doerfel, Martin Atzmueller, Andreas Hotho und Gerd Stumme: *Resource-Aware On-Line RFID Localization Using Proximity Data.* In: *ECML/PKDD (3)*, Seiten 129–144, 2011. misc = 27.

Wal. Wallace, Marc: *Radius Networks - A Note From the CEO.* `http://developer.radiusnetworks.com/2014/07/14/a-note-from-the-ceo.html`. Accessed: 2015-06-01.

WF99. Witten, Ian H. und Eibe Frank: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.